CERTIFICATE OF MAILING OR TRANSMISSION
37 C.F.R. § 1.8

Pursuant to 37 C.F.R. § 1.8, I hereby certify that I have information and a reasonable basis for belief that this paper will be filed with the United States Patent and Trademark Office via the USPTO electronic filing system (EFS) on June 26, 2007.

Signature

6-26-07
Date

PATENT
016295.0732

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:	§	
	§	
Lin, et al.	§	
	§	
Serial No.: 09/998,153	§	Group Art No. 2194
	§	
Filed: November 29, 2001	§	Examiner: Phuong N. Hoang
	§	
For: System and Method for Dynamic Device Driver Support in an Open Source Operating System	§	

AMENDMENT AND REMARKS

In response to the Final Office Action mailed February 26, 2007, Applicants submit this response and respectfully request reconsideration of the Examiner's rejections.

Amendments to the Claims

A complete list of pending claims follows, with indicated amendments:

1. (Currently Amended) A method for establishing a device driver in an open source operating system, comprising the steps of:

providing a device driver having at least one module in executable form and a service layer in open source form; and

compiling the service layer against the kernel of the open source operating system after each modification to the kernel of the open source operating system, wherein the step of compiling the service layer against the kernel comprises the step of associating the naming convention of function calls in the kernel to the naming convention of expected function calls in the device driver;

wherein the compiled service layer acts as an interface between the kernel of the operating system and the at least one executable module of the device driver.

2. (Cancelled)

3. (Original) The method for establishing a device driver in an open source operating system of claim 1, further comprising the step of linking the compiled service layer to the at least one module in executable form to form the device driver.

4. (Original) The method for establishing a device driver in an open source operating system of claim 3, further comprising the step of storing the device driver in memory.

5. (Original) The method of claim 1, wherein the step of providing a device driver comprises the step of providing a device driver having multiple modules in executable form, each of the modules associated with a hardware architecture of a computer system.

6. (Original) The method for establishing a device driver in an open source operating system of claim 5, further comprising the step of linking the compiled service layer to the multiple modules in executable form to form the device driver.

7. (Original) The method for establishing a device driver in an open source operating system of claim 6, further comprising the step of storing the device driver in memory.

8. (Currently Amended) A computer system comprising:

a processor;

a memory;

an open source operating system having a kernel;

a device driver, the device driver comprising,

an executable module compiled from an open source service layer, and

at least one executable module,

wherein the executable module compiled from the open source service layer provides an interface between the kernel of the operating system and the at least one executable module such that the executable module compiled from the open source service layer receives kernel-specific function calls from the kernel of the operating system, ~~and~~ wherein the executable module is compiled from the open source service layer following each modification

to the kernel of the operating system, and wherein the kernel of the operating system and the executable module compiled from the open source service layer send and receive function calls according to the same naming convention.

9. (Original) The computer system of claim 8, wherein the device driver is loaded in memory of the computer system.

10. (Original) The computer system of claim 8, wherein the device driver comprises multiple executable modules and wherein each of the executable modules is associated with a hardware architecture of a computer system.

11. (Cancelled)

12. (Currently Amended) The computer system of claim 8 ~~11~~, wherein the name convention comprises the use of a suffix for the naming of function calls, the suffix providing a naming convention that is specific to the kernel of the operating system.

13. (Currently Amended) A method for loading a device driver in a computer system having an open source operating system, comprising the steps of:

compiling an open source service layer against the kernel of the operating system following a modification to the kernel of the operating system; and

linking the compiled service layer to a set of precompiled driver modules, each of the precompiled driver modules being associated with a hardware architecture of a computer system;

wherein the compiled service layer provides an interface between the kernel of the operating system and the precompiled driver modules, and wherein the kernel of the operating system and the compiled service layer is operable to send and receive function calls that are named according to the same naming convention.

14. (Cancelled)

15. (Original) The method for loading a device driver of claim 13, further comprising the step of recompiling the open source service layer if it is determined that the kernel of the open source service layer has been modified.

16. (Previously Amended) The method for loading a device driver of claim 13, further comprising the step of linking the recompiled service layer to the set of precompiled driver modules.

17. (Original) The method for loading a device driver of claim 13, further comprising the step of determining, prior to compilation of the open source service layer, whether a precompiled device driver exists that is associated with the kernel of the operating system and loading the precompiled device driver if such a device driver exists.

18. (Original) The method for loading a device driver of claim 13,

wherein the function calls passed between the kernel of the operating system and the compiled open source service layer are not specific to the hardware architecture of the computer system; and

wherein the function calls passed between the compiled open source service layer and the precompiled driver modules are specific to the hardware architecture of the computer system.

19. (Previously Amended) The method for loading a device driver of claim 13, further comprising the steps of,

recompiling the service layer if it is determined that the kernel of the operating system has been modified; and

relinking the recompiled service layer to the set of precompiled driver modules.

20. (Original) The method for loading a device driver of claim 19, wherein the recompiled service layer is operable to send and receive function calls that are named according to the same naming convention.

Remarks

Claims 1, 3-10, 12-13, and 15-20 are pending in this application. Claims 2, 11, and 14 have been cancelled herein, and independent claims 1, 8, and 13 have been amended to include the limitations of the cancelled claims. The Examiner has rejected claims 1-11 and 13-20 as being obvious under 35 U.S.C. 103(a) over "Kernel Korner Writing a Linux Driver" by Matia (hereinafter "Matia") in view of "SCONE: Using Concurrent Objects for Low-level Operating System Programming" by Itoh (hereinafter "Itoh"). The Examiner has additionally rejected claim 12 under 35 U.S.C. 103(a) over Matia in view of Itoh and further in view of U.S. Patent No. 6,754,858 to Broman (hereinafter "Broman").

A. The Combination of Matia and Itoh Does Not Establish a Prima Facie Case of Obviousness as to Independent Claims 1, 8, and 13

Applicants respectfully submit that a prima facie case of obviousness has not been established and that a rejection of the pending claims on obviousness grounds is improper. A prima facie case of obviousness requires a showing that all of the claim limitations of the rejected claims are taught or suggested by the prior art. Manual of Patent Examining Procedure 2143 and 2143.03. The establishment of a prima facie case of obviousness requires that *all* the claim limitations be taught or suggested by the prior art. MPEP 2143.01 (emphasis added). "All words of a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 U.S.P.Q. 494, 496 (CCPA 1970). Here, a prima face case of obviousness is not established because (a) the combination of Matia and Itoh does not disclose or suggest the kernel and the device driver (or some portion of the device driver, such as the executable module compiled from the service layer) having naming conventions for their function calls that are associated or the same; and (b) the combination of Matia and Itoh

does not disclose or suggest the step of "compiling the service layer against the kernel . . . after each modification to the kernel."

1. The Combination of Matia and Itoh Does Not Disclose or Suggest a Naming Convention for Function Calls that is the Same for the Kernel and the Device Driver

According to the Examiner, with respect to dependent claims 2, 11, and 14 (now incorporated into independent claims 1, 8, and 13, respectively), Matia teaches associating the naming convention of function calls in the kernel to the naming convention of expected function calls in the device driver. (Office Action, p.4-5) However, the cited portion of Matia ("perform a call, page 2") does not teach or suggest associated or similar naming conventions for function calls in a kernel and device driver. At best, the cited portion of Matia discusses performing calls from shell and library functions to "a low level function of the OS." (Matia, p.2) However, nowhere does Matia discuss naming conventions of the kernel or device drivers, and specifically, Matia does not teach or suggest that the naming convention of function calls in the kernel and a device driver be associated (or the same), as required by each of the independent claims, as amended. The Examiner has not cited to Itoh as remedying this deficiency of Matia, and as such, all of the claim limitations are not taught or suggested by the prior art. For at least this reason, the Examiner's obviousness rejection of claims 1, 8, and 13 should be withdrawn.

2. The Combination of Matia and Itoh Does Not Disclose or Suggest Compiling the Driver against the Kernel after Each Modification to the Kernel

The Examiner stated in the Response to Arguments, "once the kernel calls drivers, and the driver is compiled after modification; it means the driver recompilation against the kernel and to integrate into the kernel (page 2 and 7)." (Office Action, p.8) Applicants disagree with the Examiner's assertion that Matia discloses a driver recompilation against the kernel after each modification to the **kernel**. The Examiner has only pointed to pages 2 and 7 of Matia without

referring in any way to the claim limitation of compiling the driver against the kernel after each modification to the kernel. Page 2 of Matia describes drivers generally, but does not concern driver compilation. Additionally, pages 6-7 of Matia fail to teach or suggest that the driver is compiled against the kernel after each modification to the kernel.

Matia concerns the recompilation of the device driver following a modification to the device driver. As an example, on page 8 of Matia, under the heading "Implementation of Driver Functions", the user is given instructions on "programming your own driver." These instructions continue through page 10 and concern steps for recompiling the driver following a modification *to the driver itself* and **not** to the kernel. Page 10 of Matia, for example, concerns the "task of integrating the driver into the kernel" and describes the step of "re-compile the driver." The recompilation of Matia, however, occurs after a modification to the **driver**, and **not** after each modification to the **kernel**, as required by the independent claims of the application.

Additionally, on page 11 of Matia, the user is told that it is recommended that the driver be compiled **alone** before linking the kernel. This is not the same as compiling the service layer **against the kernel,** as required by the independent claims. Additionally, Matia, on page 11, describes configuring the kernel after compiling the driver alone. Matia does not disclose the element of the independent claims that requires that the server layer be compiled against the kernel **after** each modification to the **kernel**. The Examiner does not refer to Itoh as disclosing or suggesting these elements. Because these elements of the claims are not taught or suggested by Matia in combination with Itoh, a prima facie case of obviousness cannot be established by the combination of Matia and Itoh. As such, the rejection of claims 1, 8, and 13 should be withdrawn.

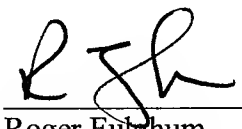
B. Dependent Claims 3-7, 9-10, 12, and 15-20

Dependent claims 3-7, 9-10, 12, and 15-20 will not be discussed individually herein, as these claims depend, either directly or indirectly, from an otherwise allowable base claim.

Conclusion

Applicants respectfully submit that the pending claims 1, 3-10, 12-13, and 15-20 of the present invention, as previously amended, are allowable. Applicants respectfully request that the rejection of the pending claims be withdrawn and that these claims be passed to issuance.

Respectfully submitted,



Roger Fulghum
Registration No. 39,678

Baker Botts L.L.P.
910 Louisiana
One Shell Plaza
Houston, Texas 77002-4995
(713) 229-1707

Baker Botts Docket Number: 016295.0732

Date: June 26, 2007